COMPUTABILIDAD: IMPLICACIONES EN LOS FUNDAMENTOS DE LAS MATEMÁTICAS

Guillermo Morales Luna

Departamento de Computación
Centro de Investigación y de Estudios Avanzados del IPN
(CINVESTAV-IPN)
México, D. F.

25 de marzo de 2010

1. RESUMEN
2. UN POCO DE HISTORIA
   - Acertijo de aritmética real
   - Algunos hechos
3. COMPUTABILIDAD
   - Propiedades básicas
   - Números computables
   - Otros problemas no-computables
   - Determinismo y no-determinismo
4. ASPECTOS FILOSÓFICOS
5. REFERENCIAS

Resumen

Desde la Grecia Clásica el constructivismo fue esencial en la investigación matemática.

Sólo hasta el S. XIX se probaron irresolubles algunos problemas antiquísimos, y hasta la primera mitad del XX se formalizó la noción de computable, así como se precisó algunas de sus limitaciones más inmediatas.

La clase de lenguajes computables admite diversas jerarquías y en ella se plantea uno de los famosos problemas del milenio: *Decidir si el determinismo y el no-determinismo de tiempo polinomial coinciden*.

Todo esto tiene implicaciones técnicas (conseguir programas eficientes) y filosóficas (controversias entre constructivismo e intuicionismo por un lado y el formalismo, por otro).

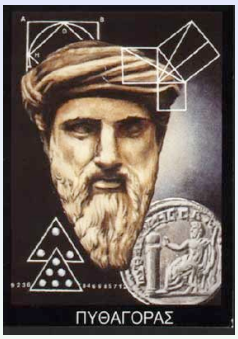
Veremos algunas de nociones involucradas en la computabilidad y presentaremos ejemplos ilustrativos, procurando omitir detalles técnicos engorrosos.

Acertijo de aritmética real

PREGUNTA

¿Existen dos racionales tales que uno elevado al otro dan un irracional?

Claro que sí:
$x = 2$, $y = x^{-1}$, $x^y = \sqrt{2}$
(y esto ya lo sabían los pitagóricos).

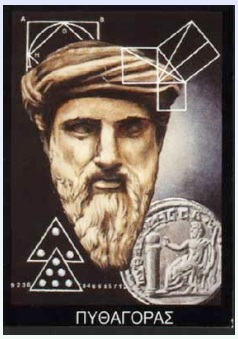


Acertijo de aritmética real

PREGUNTA

¿Existen dos racionales tales que uno elevado al otro dan un irracional?

Claro que sí:
$x = 2$, $y = x^{-1}$, $x^y = \sqrt{2}$
(y esto ya lo sabían los pitagóricos).



PREGUNTA

¿Existen dos irracionales tales que uno elevado al otro dan un racional?

Admitiendo el principio del tercero excluído puede verse que sí:

Si $\sqrt{2}^{\sqrt{2}} \in \mathbb{Q}$ **entonces**
tómese $x := \sqrt{2}$ e $y := \sqrt{2}$
en otro caso
tómese $x := \sqrt{2}^{\sqrt{2}}$ e $y := \sqrt{2}$;
dése la pareja (x, y).

Se ve que existe tal pareja pero no cuál puede ser.
Esta es una prueba de existencia meramente formal,
no es constructiva.

PREGUNTA

¿Existen dos irracionales tales que uno elevado al otro dan un racional?

Admitiendo el principio del tercero excluído puede verse que sí:

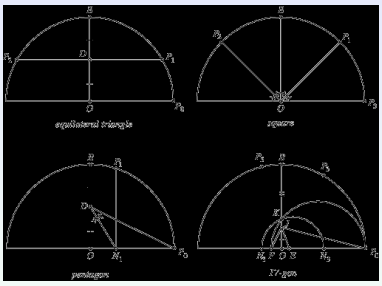
Si $\sqrt{2}^{\sqrt{2}} \in \mathbb{Q}$ **entonces**
tómese $x := \sqrt{2}$ e $y := \sqrt{2}$
en otro caso
tómese $x := \sqrt{2}^{\sqrt{2}}$ e $y := \sqrt{2}$;
dése la pareja (x, y).

Se ve que existe tal pareja pero no cuál puede ser.
Esta es una prueba de existencia meramente formal,
no es constructiva.

Algunos hechos

Euclides (S. IV AC), en sus Elementos, dió reglas de construcción de la aritmética de los racionales.

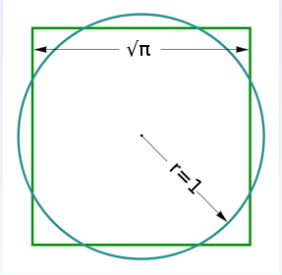
Euclides presentó asímismo construcciones con regla y compás de algunos polígonos regulares.

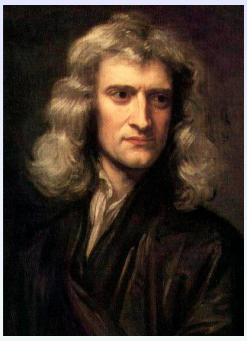


A inicios del S. XIX Gauss (1777–1855) planteó que los polígonos regulares constructibles son aquellos cuyos números de lados son el producto de una potencia de 2 y primos de Fermat ($F_n = 2^{2^n} + 1$).

En el S. XIX se probó que un número real es constructible con regla y compás si y sólo si ese número es algebraico y está en una extensión de $\mathbb{Q}$ de grado una potencia de 2.

Al ser π trascendental (Liouville, 1844), no es posible cuadrar el círculo.





En el S. XVII, Newton (1643–1727) descubrió un determinismo maravilloso con sus Leyes de la Mecánica y la Ley de Gravitación Universal, entre otras leyes físicas debidas a él.

Leibniz (1646–1716) en cambio mencionaba que dado un conjunto arbitrario de “datos” siempre existía una ley matemática a la que se ajustaran.
En efecto, dados n puntos en el plano, hay un polinomio de grado $n-1$ que pasa por ellos.
En un Discours de métaphisique aseveraba que lo importante no es localizar una ley matemática, lo importante es localizar la ley matemática más sencilla que describe a los puntos dados.

Propiedades básicas

Una ley es propiamente una función.
Naturalmente, una función puede ser dada (entre otras varias formas) como:

- una tabla de valores (argumento,valor),
- una mera definición formal,
- un procedimiento para calcularla: $x \mapsto f(x)$,
- el límite (bajo algunos criterios) de sucesiones de otras funciones,
- puntos de potencias cartesianas,
- etc.

Si se considera especificar leyes mediante procedimientos, es necesario entonces precisar qué es un procedimiento.

Procedimiento

Debe corresponder a una cadena bien formada (según reglas sintácticas definidas), sobre un alfabeto que involucre símbolos para funciones primitivas y para operadores de composición.

Esquema de programación

Un sistema consistente de un alfabeto operacional, de sus reglas sintácticas y de su semántica procedimental.

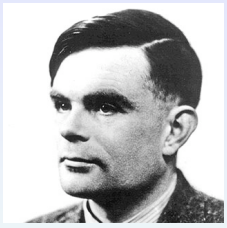
Si se considera especificar leyes mediante procedimientos, es necesario entonces precisar qué es un procedimiento.

Procedimiento

Debe corresponder a una cadena bien formada (según reglas sintácticas definidas), sobre un alfabeto que involucre símbolos para funciones primitivas y para operadores de composición.

Esquema de programación

Un sistema consistente de un alfabeto operacional, de sus reglas sintácticas y de su semántica procedimental.



Las máquinas de Turing (1912–1954) forman un esquema de programación.

Otros esquemas están dados por

- las funciones recursivas,
- los algoritmos de Markov (1903–1979),
- los lenguajes de programación convencionales,
- el cómputo cuántico,
- etc.

Una lógica formal es en sí un esquema de programación.

Si el alfabeto es finito y cada cadena bien formada también lo es, entonces la colección de procedimientos es numerable.

Una función es computable si existe un procedimiento, vale decir, un programa, que la calcula.
Así pues, inmediatamente se tiene que:

- la colección de programas es numerable, $\Pi = (\pi_n)_{n\in\mathbb{N}}$,
- la colección de funciones computables es numerable, $F = (f_m)_{m\in\mathbb{N}}$,
- para cada m existe una n tal que $f_m = f(\pi_n)$: la función calculada por el n-ésimo programa,
- dado que la colección de funciones $\mathbb{N}^* \to \mathbb{N}$ no es numerable, existen muchísimas más funciones no-computables que computables.

Tenemos así una prueba no-constructiva de la existencia de funciones no-computables.

Una función es computable si existe un procedimiento, vale decir, un programa, que la calcula.
Así pues, inmediatamente se tiene que:

- la colección de programas es numerable, $\Pi = (\pi_n)_{n\in\mathbb{N}}$,
- la colección de funciones computables es numerable, $F = (f_m)_{m\in\mathbb{N}}$,
- para cada m existe una n tal que $f_m = f(\pi_n)$: la función calculada por el n-ésimo programa,
- dado que la colección de funciones $\mathbb{N}^* \to \mathbb{N}$ no es numerable, existen muchísimas más funciones no-computables que computables.

Tenemos así una prueba no-constructiva de la existencia de funciones no-computables.

Una función es computable si existe un procedimiento, vale decir, un programa, que la calcula.
Así pues, inmediatamente se tiene que:

- la colección de programas es numerable, $\Pi = (\pi_n)_{n\in\mathbb{N}}$,
- la colección de funciones computables es numerable, $F = (f_m)_{m\in\mathbb{N}}$,
- para cada m existe una n tal que $f_m = f(\pi_n)$: la función calculada por el n-ésimo programa,
- dado que la colección de funciones $\mathbb{N}^* \to \mathbb{N}$ no es numerable, existen muchísimas más funciones no-computables que computables.

Tenemos así una prueba no-constructiva de la existencia de funciones no-computables.

Una función es computable si existe un procedimiento, vale decir, un programa, que la calcula.
Así pues, inmediatamente se tiene que:

- la colección de programas es numerable, $\Pi = (\pi_n)_{n\in\mathbb{N}}$,
- la colección de funciones computables es numerable, $F = (f_m)_{m\in\mathbb{N}}$,
- para cada m existe una n tal que $f_m = f(\pi_n)$: la función calculada por el n-ésimo programa,
- dado que la colección de funciones $\mathbb{N}^* \to \mathbb{N}$ no es numerable, existen muchísimas más funciones no-computables que computables.

Tenemos así una prueba no-constructiva de la existencia de funciones no-computables.

Una función es computable si existe un procedimiento, vale decir, un programa, que la calcula.
Así pues, inmediatamente se tiene que:

- la colección de programas es numerable, $\Pi = (\pi_n)_{n\in\mathbb{N}}$,
- la colección de funciones computables es numerable, $F = (f_m)_{m\in\mathbb{N}}$,
- para cada m existe una n tal que $f_m = f(\pi_n)$: la función calculada por el n-ésimo programa,
- dado que la colección de funciones $\mathbb{N}^* \to \mathbb{N}$ no es numerable, existen muchísimas más funciones no-computables que computables.

Tenemos así una prueba no-constructiva de la existencia de funciones no-computables.

Una función es computable si existe un procedimiento, vale decir, un programa, que la calcula.
Así pues, inmediatamente se tiene que:

- la colección de programas es numerable, $\Pi = (\pi_n)_{n\in\mathbb{N}}$,
- la colección de funciones computables es numerable, $F = (f_m)_{m\in\mathbb{N}}$,
- para cada m existe una n tal que $f_m = f(\pi_n)$: la función calculada por el n-ésimo programa,
- dado que la colección de funciones $\mathbb{N}^* \to \mathbb{N}$ no es numerable, existen muchísimas más funciones no-computables que computables.

Tenemos así una prueba no-constructiva de la existencia de funciones no-computables.

Aquí, $\mathbb{N}^* = \bigcup_{n\in\mathbb{N}} \mathbb{N}^n$ es la colección de sucesiones finitas de números naturales.

$\mathbb{N}^*$ es pues numerable y mediante una enumeración se puede identificar $\mathbb{N}^*$ con $\mathbb{N}$.

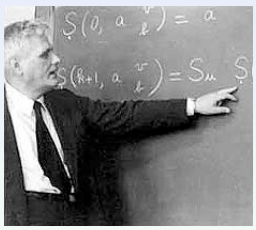
En consecuencia, cualquier entero n se puede identificar con:

- el propio entero $n \in \mathbb{N}$,
- el n-ésimo programa π_n, o
- la n-ésima función computable f_n, o
- la n-ésima sucesión finita de enteros, miembro de $\mathbb{N}^*$.

Función universal

$$U : \mathbb{N}^2 \to \mathbb{N} \ , \ (n, m) \mapsto U(n, m) = \pi_n(m) :$$

resultado en la m-ésima lista de argumentos del n-ésimo programa



Alonzo Church
(1903–1995)

Esquema de Church

Todo sistema de programación equivalente al de las máquinas de Turing: Toda máquina de Turing se puede programar en ese esquema y para todo programa hay una máquina de Turing que cacula la misma función que ese programa.

Tesis de Church. La noción de computabilidad efectiva es la realizada por cualquier esquema de Church.

En todo esquema de Church valen las propiedades siguientes:

Universalidad. Existe una función universal computable. Es decir, existe $n_U \in \mathbb{N}$ tal que

$$\pi_{n_U} : (n, m) \mapsto \pi_n(m) \tag{1}$$

Parcialidad. Hay funciones computables que no son totales. En otras palabras, hay programas que no se paran con algunas entradas.

Teorema de la recursión. Si $g : \mathbb{N} \to \mathbb{N}$ es computable y se altera con ella la enumeración de los programas, $g(\Pi) = \left(\pi_{g(n)}\right)_{n\in\mathbb{N}}$, entonces hay un índice n_m tal que π_{n_m} y $\pi_{g(n_m)}$ son equivalentes, es decir, calculan la misma función:

$$\forall m \in \mathbb{N}\ \exists n_m \in \mathbb{N}\ \forall k \in \mathbb{N} :\ \pi_{f_m(n_m)}(k) = \pi_{n_m}(k)$$

(el símbolo de igualdad aquí tiene una connotación de identidad procedimental, no de identidad sintáctica).

Autorreproducción. Existe un programa que produce una copia de sí mismo:

$$\exists n_0 \in \mathbb{N} :\ \pi_{n_0} : k \mapsto n_0.$$

El Problema de la Parada

Consideremos ahora la función que decide paradas:

$$DP : \mathbb{N}^2 \to \mathbb{N} \ , \ (n,m) \mapsto \begin{cases} 1 & \text{si } \pi_n(m) \downarrow \\ 0 & \text{si } \pi_n(m) \uparrow \end{cases}$$

Cambiando las señales de salida, se podría construir, a partir de DP, la función:

$$EP : \mathbb{N}^2 \to \mathbb{N} \ , \ (n,m) \mapsto \begin{cases} \uparrow & \text{si } \pi_n(m) \downarrow \\ 0 & \text{si } \pi_n(m) \uparrow \end{cases}$$

Definamos

$$R : \mathbb{N} \to \mathbb{N} \ , \ n \mapsto R(n) = EP(n,n)$$

Entonces

$$\forall n \in \mathbb{N} : \ [R(n) \downarrow \iff \pi_n(n) \uparrow] .$$

Si acaso R fuese computable, se tendría:

$$\exists n_0 \in \mathbb{N} \ \forall n \in \mathbb{N} : \ [\pi_{n_0}(n) \downarrow \iff \pi_n(n) \uparrow] ,$$

lo cual es contradictorio al tomar $n = n_0$.

Teorema (Incomputabilidad del Problema de la Palabra)

En todo esquema de Church, la función DP que decide paradas no es computable.

Tratados y antologías clásicos en este tema son [3, 4].

Números computables

Sea $b \in \mathbb{N}$, $b > 1$, un entero que sirva como base de representación: $b = 10$ era lo usual, $b = 2$ es convencional. Salvo omisión de "colas de $(b-1)$'s":
$\forall x \in \mathbb{R}^+$,
$\exists! n_x \in \mathbb{Z}, (a_{n,x})_{n\in\mathbb{Z}} \in [\![0, b-1]\!]^{\mathbb{Z}}$:

$$x = \sum_{n\in\mathbb{Z}} a_{n,x} b^n \;\&\; [n > n_x \;\Rightarrow\; a_{n,x} = 0]\,.$$

Un número real no-negativo $x \in \mathbb{R}$ se dice ser computable si la función $\mathbb{Z} \to [\![0, b-1]\!] \subset \mathbb{N}$, $n \mapsto a_{n,x}$, lo es.
Un real negativo x es computable si $-x$ lo es.

- Todo número racional es computable.
- π y e son computables.
- Sólo hay una cantidad numerable de números computables.
- La gran mayoría de los números irracionales no son computables.

Sea *Cmp* la colección de números reales computables.
Entonces:

- *Cmp* es un campo, es una extensión de $\mathbb{Q}$ y es un subcampo de $\mathbb{R}$.
- Existe un conjunto numerable $R \subset \mathbb{R}$ tal que $Cmp = \mathbb{Q}[R]$.
- *Cmp* es un espacio topológico metrizable con la topología usual inducida por $\mathbb{R}$.
- Toda aritmética de punto flotante está contenida en *Cmp*.

Una función real $f : \mathbb{R} \to \mathbb{R}$ es computable si $f(Cmp) \subseteq Cmp$.
Sea $U_C = [0,1] \cap Cmp$. Entonces:

- U_C posee la topología de Cantor, es decir, la topología producto en $\{0,1\}^{\mathbb{N}}$.
- Toda función computable es contínua.
- Existe una versión computable del teorema de Heine-Borel.
- El análisis real se traduce, aunque no en su totalidad, a un análisis similar en *Cmp*.

Un curso muy recomendable sobre Análisis Computable es [5]. En línea está, del mismo autor, [6] donde se puede ver las técnicas de análisis en el contexto de computabilidad.

Otros problemas no-computables

Sea $Pr : \mathcal{P}(\mathbb{N}) \to [0,1]$, $A \mapsto Pr(A) = \sum_{a \in A} 2^{-(a+1)}$. Esta es una medida de probabilidad.

PROPOSICIÓN

Sea $\Omega = \{n \in \mathbb{N} \mid \pi_n(n) \downarrow\}$ el conjunto de índices de programas que se paran con su índice, y
sea $\omega = Pr(\Omega)$.
Entonces $\omega \notin Cmp$.

En efecto, esto resulta de la irresolubilidad del problema de la parada. □

Así se tiene que aunque se haya calculado los primeros n bits en ω, no se podría tener certeza alguna sobre cuál ha de ser el siguiente bit.

Desde los años 60, Chaitin (1947–) ha estudiado ω [1, 2].

Un conjunto de puntos será tanto más determinado cuanto sea más corta la ley que lo describe. Será tanto más azaroso cuanto sea más larga la ley que lo describe.

ω es pues lo más azaroso posible porque una descripción de ella sólo puede consistir de una transcripción de ella misma. Es irreducible.

Digamos que un programa es elegante si es uno de la menor longitud entre todos los que son equivalentes a él.
Entonces, por el Principio del Buen Orden, toda función computable posee un programa elegante que la calcula.

PROPOSICIÓN

No se puede determinar cuándo se tiene un programa elegante.

Digamos que un programa es elegante si es uno de la menor longitud entre todos los que son equivalentes a él.
Entonces, por el Principio del Buen Orden, toda función computable posee un programa elegante que la calcula.

PROPOSICIÓN

No se puede determinar cuándo se tiene un programa elegante.

En efecto, supongamos por un momento que la función $E : \mathbb{N} \to \{0,1\}$, [$E(n) = 1 \Leftrightarrow \pi_n$ es elegante], sea computable.
Sea $F : \mathbb{N} \to \mathbb{N}$,
$n \mapsto F(n) =$ Arg Min$\{m > n | \text{ long}(\pi_m) > \text{long}(\pi_n)$ & $E(m) = 1\}$
y sea $G : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $(n,m) \mapsto G(n,m) = U(F(n),m)$.
Entonces G ha de ser también computable.
Sea $n_G \in \mathbb{N}$ un índice de G: π_{n_G} calcula G.
Sea $n_0 = F(n_G + k)$. Entonces $\pi(n_0)$ debe ser elegante, equivalente a $m \mapsto G(n_G + k, m)$, pero éste último es más corto que $\pi(n_0)$ para una elección adecuada de k.
Esta contradicción muestra que E no puede ser computable. □

CONSECUENCIA TÉCNICA

No existe un procedimiento general para optimizar programas.
La optimización de programas ha de ser un trabajo artesanal.

En efecto, supongamos por un momento que la función $E : \mathbb{N} \to \{0,1\}$, [$E(n) = 1 \Leftrightarrow \pi_n$ es elegante], sea computable.
Sea $F : \mathbb{N} \to \mathbb{N}$,
$n \mapsto F(n) =$ Arg Min$\{m > n | \text{ long}(\pi_m) > \text{long}(\pi_n)$ & $E(m) = 1\}$
y sea $G : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, $(n,m) \mapsto G(n,m) = U(F(n),m)$.
Entonces G ha de ser también computable.
Sea $n_G \in \mathbb{N}$ un índice de G: π_{n_G} calcula G.
Sea $n_0 = F(n_G + k)$. Entonces $\pi(n_0)$ debe ser elegante, equivalente a $m \mapsto G(n_G + k, m)$, pero éste último es más corto que $\pi(n_0)$ para una elección adecuada de k.
Esta contradicción muestra que E no puede ser computable. □

CONSECUENCIA TÉCNICA

No existe un procedimiento general para optimizar programas.
La optimización de programas ha de ser un trabajo artesanal.

Determinismo y no-determinismo

En general, un problema es un subconjunto $A \subset \mathbb{N} \times \mathbb{N}$. Si $(a, x) \in A$ se dice que a es una instancia y x una solución correspondiente.
Dado $A \subset \mathbb{N}^2$ se tiene los problemas:

DE BÚSQUEDA: dada la instancia $a \in \mathbb{N}$ encuéntrese una solución $x \in \mathbb{N}$: $(a, x) \in A$;

DE DECISIÓN: dada una pareja $(a, x) \in \mathbb{N}^2$, verifíquese si acaso x es una solución de la instancia $a \in \mathbb{N}$: $(a, x) \in A$.

PROGRAMA RESOLVEDOR es uno que dada la instancia a produce una correspondiente solución x, es decir, resuelve el problema de búsqueda.

PROGRAMA COMPROBADOR es uno que dada la instancia a genera de manera indeterminada una solución hipotética x y luego comprueba si acaso $(a, x) \in A$.

- Un programa resolvedor es exitoso si genera una solución toda vez que exista una, o bien indica que no hay soluciones en otro caso.
- Un programa comprobador es exitoso si existe una solución hipotética generada que es, en efecto, una solución verdadera.

Ejemplos

FACTORIZACIÓN. Sea $A = \{(n, m) | \ 1 < m < n \ \& \ m|n\}$.
El problema de búsqueda consiste en encontrar un factor m de un entero n.
El problema de decisión consiste en verificar si acaso $m|n$ dados m y n.

LOGARITMO DISCRETO. Sea $\mathbb{G} = \langle g \rangle$ un grupo cíclico y $g \in \mathbb{G}$ un generador. Sea $A = \{(h, n) \in \mathbb{G} \times \mathbb{N} | \ g^n = h\}$.
El problema de búsqueda consiste en calcular el logaritmo $n \in \mathbb{N}$ dado el elemento $h \in \mathbb{G}$.
El problema de decisión consiste en comprobar si acaso $h = g^n$ en $\mathbb{G}$.

- La clase P consiste de los problemas que poseen un resolvedor que realiza una cantidad de cálculos de funciones primitivas polinomial en términos de la longitud de la entrada.
- La clase NP consiste de los problemas que poseen un comprobador que realiza una cantidad de cálculos de funciones primitivas polinomial en términos de la longitud de la entrada.

Evidentemente P $\subseteq$ NP.
Uno de los problemas del Milenio consiste en decidir si acaso NP $\subseteq$ P.

Clay Mathematics Institute http://www.claymath.org/millennium/P_vs_NP/

Clay Mathematics Institute
Dedicated to increasing and disseminating mathematical knowledge

HOME | ABOUT CMI | PROGRAMS | NEWS & EVENTS | AWARDS | SCHOLARS | PUBLICATIONS

P vs NP Problem

Suppose that you are organizing housing accommodations for a group of four hundred university students. Space is limited and only one hundred of the students will receive places in the dormitory. To complicate matters, the Dean has provided you with a list of pairs of incompatible students, and requested that no pair from this list appear in your final choice. This is an example of what computer scientists call an NP-problem, since it is easy to check if a given choice of one hundred students proposed by a coworker is satisfactory (i.e., no pair taken from your coworker's list also appears on the list from the Dean's office), however the task of generating such a list from scratch seems to be so hard as to be completely impractical. Indeed, the total number of ways of choosing one hundred students from the four hundred applicants is greater than the number of atoms in the known universe! Thus no future civilization could ever hope to build a supercomputer capable of solving the problem by brute force; that is, by checking every possible combination of 100 students. However, this apparent difficulty may only reflect the lack of ingenuity of your programmer. In fact, one of the outstanding problems in computer science is determining whether questions exist whose answer can be quickly checked, but which require an impossibly long time to solve by any direct procedure. Problems like the one listed above certainly seem to be of this kind, but so far no one has managed to prove that any of them really are so hard as they appear, i.e., that there really is no feasible way to generate an answer with the help of a computer. Stephen Cook and Leonid Levin formulated the P ?= NP (i.e., easy to find) versus NP (i.e., easy to check) problem independently in 1971.

The Millennium Problems
Official Problem Description — Stephen Cook
Lecture by Vijaya Ramachandran at University of Texas (video)
Hi res essay

› *Return to top*

Contact | Search | Terms of Use | © 2010 Clay Mathematics Institute

Un problema es difícil en una clase si todos los problemas en esa clase se reducen a ese problema.

- En la actualidad no se conoce algoritmos polinomiales para resolver los problemas de búsqueda de Factorización y del Logaritmo Discreto.
- Tampoco se ha visto que sean difíciles-NP.

Sin embargo están en el centro de la seguridad en las comunicaciones actuales: hacen robustos a los sistemas criptográficos de clave pública más utilizados.

La solidez de éstos está basada pues en la confianza pública, no en una demostración matemática.

Un problema es difícil en una clase si todos los problemas en esa clase se reducen a ese problema.

- En la actualidad no se conoce algoritmos polinomiales para resolver los problemas de búsqueda de Factorización y del Logaritmo Discreto.
- Tampoco se ha visto que sean difíciles-NP.

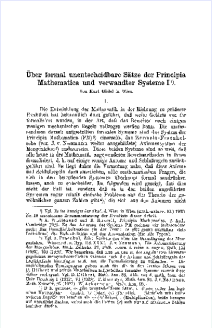
Sin embargo están en el centro de la seguridad en las comunicaciones actuales: hacen robustos a los sistemas criptográficos de clave pública más utilizados.

La solidez de éstos está basada pues en la confianza pública, no en una demostración matemática.

Aspectos filosóficos

En Lógica, la noción de computabilidad corresponde a decidibilidad: Una teoría es decidible si existe un procedimiento para reconocer cuándo un enunciado, es decir una fórmula bien formada sin variables libres, es un teorema de la lógica.

Por ejemplo, una teoría fuertemente consistente (o sea, una en la que sus modelos son elementalmente equivalentes), es decidible: Dado un enunciado simplemente se evalúa en un modelo computable.



Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I.

La artimética de Peano no es decidible.
De hecho es incompleta y además no puede demostrar su propia consistencia.

Los teoremas de incompletitud demostraron que no es viable el Programa de Hilbert (1862–1943):
Demostrar la consistencia de la matemática en la misma matemática, mediante pruebas "formales".

De hecho, parafraseando a Chaitin, la existencia de números no-computables hace implausible que exista una *"teoría matemática de (o para) todo"*.
Por tanto, *acaso la matemática ha de ser una ciencia experimental, más que basarse en meras demostraciones.*
Brouwer (1881–1966), como creador del intuicionismo planteaba que *no existen verdades matemáticas sin una componente experimental.*

- En la lógica intuicionista
  - se rechaza el principio del tercero excluído y
  - las demostraciones de existencia han de ser constructivas.
- En la computabilidad, prevalece el enfoque intuicionista sobre el formalista.
- En contraposición, acaso más allá del formalismo, hay quienes plantean que las matemáticas son creaciones del cerebro humano (aunque) también son efectivas en el universo.
  Así que con tal enfoque prevalece el aspecto puramente demostrativo.

Referencias

- Gregory Chaitin. *The Limits of Mathematics : A Course on Information Theory and the Limits of Formal Reasoning*. Springer, October 2002.
- Gregory Chaitin. *Meta Math!: The Quest for Omega*. Vintage, November 2006.
- Martin Davis. *The Undecidable*. Dover Publications, Incorporated, 2004.
- Hartley Rogers, Jr. *Theory of recursive functions and effective computability*. MIT Press, Cambridge, MA, USA, 1987.
- Klaus Weihrauch. *Computable Analysis*. Springer, November 2000.
- Klaus Weihrauch and Ning Zhong. Computable analysis of the abstract Cauchy problem in a Banach space and its applications (I). *Electron. Notes Theor. Comput. Sci.*, 167:33–59, 2007.

¡¡Gracias por su atención!!

¿Preguntas?

gmorales@cs.cinvestav.mx
http://delta.cs.cinvestav.mx/~gmorales